Optical Self-Adjusting Data Center Networks in the Scalable Matching Model

Caio A. Caldeira, Otavio A. de O. Souza, Olga Goussevskaia, and Stefan Schmid

Abstract—Self-Adjusting Networks (SAN) optimize their physical topology toward the demand in an online manner. Their application in data center networks is motivated by emerging hardware technologies, such as 3D MEMS Optical Circuit Switches (OCS). The Matching Model (MM) has been introduced to study the hybrid architecture of such networks. It abstracts from the electrical switches and focuses on the added (reconfigurable) optical ones. MM defines any SAN topology as a union of matchings over a set of top-of-rack (ToR) nodes, and assumes that rearranging the edges of a single matching comes at a fixed cost.

In this work, we propose and study the Scalable Matching Model (SMM), a generalization of the MM, and present OpticNet, a framework that maps a set of ToRs to a set of OCSs to form a SAN topology. We prove that OpticNet uses the minimum number of switches to realize any bounded-degree topology and allows existing SAN algorithms to run on top of it, while preserving amortized performance guarantees. Our experimental results based on real workloads show that OpticNet is a flexible and efficient framework for the implementation and evaluation of SAN algorithms in reconfigurable data center environments.

Index Terms—Reconfigurable Data Center Networks, Self-Adjusting Networks, Optical Circuit Switching, Matching Model

1 Introduction

The design of efficient data center networks has received significant attention in recent years [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], fueled by data-centric online applications, such as web search, social networks, and multimedia. Empirical studies show that communication patterns in data centers feature much spatial and temporal locality [5], [6], [21], i.e., traffic is bursty and traffic matrices skewed and clustered. This structure represents an untapped potential for building more efficient communication networks. This is the advantage of demand-aware topologies, based on, e.g., 3D MEMS optical circuit switches (OCS) [4], [5], [6], [7], [8], [9], [10], [22], [23], [24], [25], which can provide shortcuts to elephant flows. New hardware technologies are being developed, such as soliton microcombs [22] and tunable lasers [1], offering increasingly faster (sub-nanosecond) switching time. In this context, scheduling or matching algorithms [3], [6], [19], [20], [26], [27], [28] can determine the following configuration based on the network state, and the network topology can be reconfigured on-demand in real-time. A few prototypes based on commodity-off-the-shelf OCS have been built, and their advantages have been demonstrated, e.g., [2], [8], [25]. There are also first deployments, e.g. by Google [13], [29], [30].[1]

Self-Adjusting Networks (SAN) [5] dynamically optimize their physical topology based on real-time demand, without prior knowledge of traffic patterns. When analyzing the performance of SANs, it is necessary to distinguish between *adjustment cost* and *service cost*. The former refers to the cost of network reconfiguration (e.g., energy, latency, and control plane overhead due to physical network topology adjustments), and the latter refers to the price of serving each communication request (e.g., the delay proportional to source-destination route length). Most of the existing SAN algorithms [7], [27], [28] have been based on *edge-distance* adjustment cost models, which define adjustment cost as the number of edges replaced between consecutive network topologies. This is a useful basic model that enabled the first algorithmic results. In practice, however, switching hardware usually allows to reconfigure the topology at a per-matching granularity, i.e., the adjustment cost is proportional to the number of OCS reconfigurations, where each switch internally connects its in$\rightarrow$out ports via a matching[2].

ToR-Matching-ToR (TMT) [31] is a two-layer architecture for data center networks, in which each top-of-rack (ToR) is represented by a (leaf) static packet switch, and n ToRs are connected by a set of (spine) reconfigurable OCSs. Each spine switch internally connects n in$\rightarrow$out ports via a matching that may be reconfigured at runtime. TMT can be used to model existing systems, such as RotorNet [12], Opera [11], ProjecToR [6], Cerberus [32], and others [19]. The Matching Model (MM) [31] was introduced to study such TMT architectures. It abstracts from the electrical (static) switches and focuses on the added (reconfigurable) optical ones. MM assumes that any (hybrid) network topology can be defined as a union of matchings over the set of nodes and that rearranging the edges of a single matching comes at a fixed cost. Thus, the total *adjustment cost* for adjusting the

• *C.A. Caldeira, O.A.O.Souza and O. Goussevskaia are with the Computer Science Department, Universidade Federal de Minas Gerais (UFMG), 31270-901 Belo Horizonte, Brazil.*
E-mails: caio.caldeira@dcc.ufmg.br, oaugusto@dcc.ufmg.br, olga@dcc.ufmg.br
• *Stefan Schmid is with the Faculty of Computer Science, TU Berlin, 10587 Berlin, Germany.*
E-mail: stefan.schmid@tu-berlin.de

1. This work was supported by CAPES, CNPq, Fapemig, and the European Research Council (ERC), #864228 (AdjustNet), 2020-2025.
2. A matching is defined as a set of non-adjacent edges in a graph.

whole topology to a new one is determined by the number of matchings needed to construct the topology. Early work has shown this model's relevance, e.g. in [33] the authors simulate several so-called lazy SANs in the MM, based on spaced out (less frequent) calls of existing (sequential) SANs, such as SplayNets [7] and ReNets [34].

In this work, we propose the Scalable Matching Model (SMM), which generalizes the MM by adding the constraint that the number of ports of each OCS switch is constant while relaxing the constraint that the number of OCS switches is constant. SMM adds flexibility to the network design, depending on the cost and constraints on available switching hardware. SMM uses larger leaf-layer (static) electrical switches, which typically come at a lower per-port hardware price than OCS hardware [35], and a greater number of smaller spine-layer (optical) switches, which can be incrementally appended to the existing architecture (even at runtime) in case new ToRs are added to the data center.

Furthermore, we present and evaluate OpticNet, a framework that maps a set of ToRs to a set of OCSs. We prove that OpticNet uses a minimal number of reconfigurable switches to realize any desired network topology and allows to execute existing SAN algorithms on top of it while preserving the performance (asymptotic cost) guarantees of the latter in the SMM.

Summary of contributions and organization: In Section 2, we introduce the Scalable Matching Model (SMM), a generalization of the Matching Model (MM) [31], that allows to model TMT architectures in which the number of ToRs (arbitrarily) exceeds the number of in-out ports of the spine-layer switches. In Section 3, we describe the general architecture of OpticNet, a framework that maps a set of (leaf-layer) ToR switches to a set of (spine-layer) demand-aware reconfigurable OCSs to form any bounded-degree network topology. In Section 4 we describe the algorithmic and implementation details of OpticNet. In Section 5 we prove that OpticNet is correct and optimal in the number of spine switches, and that any SAN algorithm can run on top of it while preserving amortized performance guarantees in the SMM. In Section 6 we describe the implementation of OpticNet and evaluate its performance in combination with state-of-the-art SAN algorithms. Finally, in Section 7, we discuss related work, and in Section 8, we present our conclusions.

2 Model

Traffic and SANs: The objective of SAN algorithms is to create a network topology to connect a set V of n communication nodes (ToRs, top-of-rack electrical switches). The input to the problem is given by a sequence σ of m messages $\sigma_i(s_i, d_i) \in V \times V$ occurring over time, with source s_i and destination d_i; m can be infinite. We denote by b_i the time when a message σ_i is generated and by e_i the time it is delivered. The sequence σ is revealed over time, in an online manner, and can be arbitrary (e.g., chosen adversarially). When serving these communication requests, the network can adjust over time through a sequence of network topologies $N_0, N_1, \ldots$. Each N_t should belong to some desired graph family $\mathcal{N}$. The networks should be of *bounded degree* for scalability reasons.

Online reconfigurations: To minimize the communication cost and adjust the topology smoothly over time, the network is reconfigured (locally) through adjustments that preserve the desired properties of the graph family $\mathcal{N}$. SANs like SplayNet [7] and DiSplayNet [36] are based on Binary Search Trees (BST) and extend the classical *zig, zig-zig* and *zig-zag* rotations, first introduced for splay trees [37]. CBTrees [38] and CBNet [27] leverage *bottom-up* and *top-down* semi-splaying (*semi-zigzig* and *semi-zigzag*). Each splay operation updates a constant number of links at a constant cost and roughly halves the depth of every node along the communication path. This halving effect makes splaying efficient in an amortized sense.

TMT [31] is a useful architecture to model reconfigurable datacenter networks (RDN). It is a two-layer network in which a set of *static* leaf-layer (ToR) switches is connected using a set of *reconfigurable* spine-layer switches (OCS). Each OCS internally connects its in-out ports via a *matching*, which can be dynamic and change over time. The matching model [31] is a formalization of the TMT architecture. The network consists of a set of n nodes (leaf-layer static electrical switches) are connected using $k' = O(1)$ (spine-layer reconfigurable optical) switches, $SW = \{sw_1, sw_2, \ldots, sw_{k'}\}$, and each switch internally connects its n *in$\rightarrow$out ports* via a *matching*. At each time t, the network is the union of these matchings: $N_t = \mathcal{M}_t = \cup_{i=1}^{k'} M(i,t)$, where $M(i,t)$ denotes the matching on switch sw_i at time t.

SMM: In this work, we propose the Scalable Matching Model, which extends the MM by adding the constraint that the number of ports of each spine-layer switch is constant while relaxing the constraint that the number of spine switches is constant. Let $\mathcal{N}$ be the family of **bounded degree ($\leq$ k) graphs** on a set V of n nodes (leaf-layer switches) and $\mathcal{N}_t \in \mathcal{N}$ be a network topology at time t. Let $SW = \{sw_1, \ldots, sw_{n'}\}$ be a set of (spine-layer) switches of size $n' = f(n, p, k)$, where each $sw_i \in SW$ internally connects its $p = O(1)$ in$\rightarrow$out ports via a *matching* $M(i,t)$ of size $\leq p$. As in MM, at each time t, the network N_t is equal to the union of all these matchings:

$$N_t = \mathcal{M}_t = \cup_{i=1}^{n'} M(i,t).$$

Port connections: The port-to-port physical connections between leaf and spine layers in a TMT architecture can be realized differently, according to the hardware specification (e.g., via free-space optics [6]). In our simulations, we assume that leaf-spine connections are static *simplex* links[3], i.e., for two ToRs (leaf-layer switches) $(u,v) \in N_t$ to establish a connection, we need that $\exists sw_x \in SW | (u,v) \in M(x,t)$. Moreover, there must be a static (physical) link connecting an output port of u to an input port (in_x^u) of sw_x and a static link from an output port (out_x^v) of sw_x to an input port of v.

Before the system (RDN) starts operating, the TMT architecture must be configured according to these requirements, i.e., we need to find a one-to-one assignment, each representing a simplex link, between all in$\rightarrow$out and out$\rightarrow$in ports of leaf-spine switch pairs.

In the MM, the port mapping between leaf-layer and spine-layer switches is straightforward: each ToR $i, 1 \leq i \leq$

3. Note that a duplex communication can be converted to simplex, and vice-versa, by duplicating each duplex switch or removing copies of simplex switches, resp.

n has k uplinks, where uplink $j, 1 \le j \le k$ connects to a port of index i of (the spine) switch sw_j. The directed outgoing (leaf) uplink is connected to the incoming port (in^i_j) of sw_j, and the directed incoming (leaf) uplink is connected to the outgoing port (out^i_j) of sw_j. Each switch has n input ports and n output ports. Its connections are directed from input to output ports. At any point in time, each switch $sw_j \in SW$ provides a matching of size $\le n$ between its input and output ports. In SMM, however, because spine-layer switches are limited to a constant number of in$\to$out ports, the problem of connecting leaf-layer and spine-layer switches has a different structure.

Port Mapping Problem (SMM-PMP): Let $\mathcal{N}$ be the family of bounded-degree ($\le k$) graphs[4] on a set V of n nodes (static leaf-layer ToR switches), and $SW = \{sw_1, \dots, sw_{n'}\}$ be a set of $n' = f(n, p, k)$ (reconfigurable spine-layer) switches, where each switch $sw_i \in SW$ internally connects its in$\to$out port-pairs via a *matching* of size $\le p$.

Let $\mathcal{P}(sw_x) \subseteq V \times V, |\mathcal{P}(sw_x)| = p^2$ be the in$\to$out port-set of an OCS switch $sw_x \in SW$. When clear from the context, we will use the notation $(u, v) \in \mathcal{P}(sw_x)$ to refer to an in$\to$out port-pair mapped to switch sw_x. When needed to distinguish between node (ToR) and (switch) port identifiers, we will use the notation $(in^u_x, out^v_x) \in \mathcal{P}(sw_x)$, where $(u, v) \in V \times V$.

Given a network topology $N_t \in \mathcal{N}$ at time t, the objective is to find a set of switches SW, a set of matchings $\mathcal{M}_t = \cup_{sw_x \in SW} M(x, t) = N_t$, and all in$\to$out port sets $\mathcal{P}(sw_x), sw_x \in SW$, such that $\forall (u, v) \in N_t$:

$$\exists sw_x \in SW |\ (u, v) \in M(x, t) \text{ and } M(x, t) \subset \mathcal{P}(sw_x).$$

If switches are not duplex and edges are undirected, we also need to insert the symmetric directed edge (v, u):

$$\exists sw_y \in SW |\ (v, u) \in M(y, t) \text{ and } M(y, t) \subset \mathcal{P}(sw_y).$$

Time model: In sequential SAN algorithms [7], [33], [37], each message σ_i is processed strictly after the previous one σ_{i-1} has been delivered. In this work, we consider a concurrent execution scenario. We assume that time is divided into synchronous rounds, and each round consists of a constant number of time-slots. In a round, multiple (independent) nodes can make adjustments concurrently. We consider that nodes and communication between them are reliable.

Consider a sequence σ of m messages, a SAN algorithm $\mathcal{A}$, any initial network $N_0 \in \mathcal{N}$, and a message $\sigma_i(s_i, d_i) \in \sigma$, generated at time b_i and delivered at time e_i. In terms of time, one objective is to minimize the makespan: $Makespan(\mathcal{A}, N_0, \sigma) = \max_{1 \le i \le m} e_i - b_1$.

Communication cost: Let us define the *adjustment cost* adj_i as the number of adjustments performed by $\mathcal{A}$ to deliver σ_i and the *service cost* srv_i to be the price of routing the message along the (shortest) path $dist_{N_{e_i}}(s_i, d_i), N_{e_i} \in \mathcal{N}$. We assume that routing a message incurs a cost of 1 unit per hop. MM allows to define the adjustment cost in two different ways:

Link updates: In this case, adjustment cost is proportional to the number of different edges between a pair of consecutive matchings on the same switch $sw_x \in SW$. If the cost of a single edge is α, the adjustment cost for the entire network is defined as:

$$linkAdj(N_{t-1}, N_t) = \alpha \sum_{sw_x \in SW} |M(x, t) \setminus M(x, t-1)|.$$

Switch updates: In this case, if a matching (switch) is reconfigured (adjusted), it costs α regardless of the number of edge changes in the matching. Let $\mathbb{I}_{S=S'}$ be an indicator function that denotes if set S is equal to set S'. Then, the adjustment cost for the network is:

$$swAdj(N_{t-1}, N_t) = \alpha \sum_{sw_x \in SW} \mathbb{I}_{M(x,t)=M(x,t-1)}.$$

Note that we always have that $linkAdj(N_{t-1}, N_t) \ge swAdj(N_{t-1}, N_t)$. We distinguish between link ($linkAdj$) and switch ($swAdj$) adjustment costs for the following reason. Depending on the particular OCS hardware specification, it may or may not be possible to update an individual link (in$\to$out port mapping) without interrupting all communication links on the switch to which that link belongs. When individual link reconfiguration is enabled by the OCS hardware, the $linkAdj$ definition of adjustment cost is more accurate. However, if topology reconfigurations are executed at a per-matching granularity, the $swAdj$ metric is more appropriate.

Let $t_{\max} = \max_{1 \le i \le m} e_i$, where e_i is the time (round) when a message $\sigma_i(s_i, d_i)$ has been delivered to its destination node d_i, and $dist_{N_{e_i}}(s_i, d_i)$ be the hop distance between the source and the destination in N_t at time $t = e_i$. Finally, we define the **total** service cost, total link and switch adjustment costs, and the total link and switch (communication) costs, respectively, as follows:

$$\begin{aligned}
\text{Srv}(\mathcal{A}, N_0, \sigma) &= \sum_{i=1}^{m} (dist_{N_{e_i}}(s_i, d_i) + 1), && (1)\\
\text{LiAdj}(\mathcal{A}, N_0, \sigma) &= \sum_{t=1}^{t_{\max}} linkAdj(N_{t-1}, N_t), && (2)\\
\text{SwAdj}(\mathcal{A}, N_0, \sigma) &= \sum_{t=1}^{t_{\max}} swAdj(N_{t-1}, N_t), && (3)\\
\text{LiCost}(\mathcal{A}, N_0, \sigma) &= \text{Srv}(\mathcal{A}, N_0, \sigma) + \text{LiAdj}(\mathcal{A}, N_0, \sigma), && (4)\\
\text{SwCost}(\mathcal{A}, N_0, \sigma) &= \text{Srv}(\mathcal{A}, N_0, \sigma) + \text{SwAdj}(\mathcal{A}, N_0, \sigma). && (5)
\end{aligned}$$

3 OPTICNET ARCHITECTURE

OpticNet is a framework for implementing SAN algorithms in the TMT network architecture. In this section we describe how OpticNet solves the SMM-PMP problem.

Consider a network topology $N_t \in \mathcal{N}, t \ge 0$ of bounded degree ($\le k$) on a set V of n vertices, with distinct identifiers in $[1, n]$. We assume that each node (ToR) $u \in V$ has a (leaf-layer) static packet switch on its top. We construct a set $SW = \{sw_1, \dots, sw_{n'}\}$ of n' (spine-layer) OCS switches, such that each $sw_x \in SW$ internally connects p in$\to$out port pairs via a matching (a set of non-adjacent edges) $M(x, t)$ of size $\le p$. For simplicity, we assume that n is a multiple of p (or, equivalently, that $1 \le x < p$ dummy vertices are added to the network).

4. W.l.g., we assume that links are undirected and unweighted.

First, we partition V into $C = \lceil n/p \rceil$ clusters $C_i \subseteq V, |C_i| = p$, as follows: $\forall (u,v) \in V \times V$, if $u < v$ then $u \in C_i, v \in C_j$, where:

$$i = \min\left(\left\lfloor \frac{u}{p} \right\rfloor, \left\lfloor \frac{v}{p} \right\rfloor\right),\ j = \max\left(\left\lfloor \frac{u}{p} \right\rfloor, \left\lfloor \frac{v}{p} \right\rfloor\right). \tag{6}$$

Next, we partition the set of OCS switches into two subsets ($SW = SW^{\cup} \cup SW^{\times}$), by type: *unit switches* ($SW^{\cup}$) and *cross switches* ($SW^{\times}$), defined as follows:

Unit switch: Given a cluster of nodes (ToRs) $C_i, 0 \le i < C$, the in$\rightarrow$out port-set of a switch $sw_x \in SW^{\cup}$ of type *unit* is defined as $\mathcal{P}(sw_x) = C_i \times C_i,\ sw_x \in SW^{\cup}$.

Cross switch: Given a pair of clusters $(C_i, C_j), 0 \le i < j < C$, the in$\rightarrow$out port-set of a switch $sw_x \in SW^{\times}$ of type *cross* is defined as $\mathcal{P}(sw_x) = C_i \times C_j,\ sw_x \in SW^{\times}$.

Therefore, a unit-switch connects nodes that belong to the same cluster, and a cross-switch interconnects nodes that belong to two distinct clusters. Note that in the case $(p = n)$ there is only one cluster and, therefore, no cross-switches.

Each cluster $C_i, 0 \le i < C$, will be assigned to a subset of k switches of type *unit* ($\subseteq SW^{\cup}$), and each pair of clusters $(C_i, C_j), i \ne j$ will be assigned to a subset of $2k$ switches of type *cross* ($\subseteq SW^{\times}$). We define the in$\rightarrow$out port-set for a switch sw_{idx} of index idx according to its type, as follows:

$$\mathcal{P}(sw_{idx}) \quad = \quad \begin{cases} C_i \times C_i, \quad \forall sw_{idx} \in SW^{\cup} \mid 0 \le i < C, \\ \quad idx = \theta(i,j) + x,\ \ 0 \le x < k, \\ C_i \times C_j, \forall sw_{idx} \in SW^{\times} \mid 0 \le i < j < C, \\ \quad idx = \theta(i,j) + x,\ \ 0 \le x < 2k, \end{cases}$$

where the *cluster-offset* $\theta(i,j), 0 \le i \le j < C$ is defined as:

$$\theta(i,j) \quad = \quad \begin{cases} ik, \ \text{ if } (i = j) \ \ \text{(unit cluster)}, \\ Ck{+}2k\sum_{x=0}^{\max(0,i-1)}(C - x - 1) \\ \quad + 2k(j - i - 1) = \\ Ck{+}ik(2C - i - 1) \\ \quad + 2k(j - i - 1), \ \ \text{otherwise}. \end{cases} \tag{7}$$

Edge indexing: To determine through which switch a (ToR) node will forward its messages, we have to know which switch each edge belongs to. Grouping the switches of each cluster within an array allows us to identify which switch an edge belongs to by assigning, to each edge $(u,v) \in N_t$, $u \in C_i$, $v \in C_j$, $0 \le i \le j < C$, a *switch-index*, defined as follows:

$$sw\_idx(u,v,p) \quad = \quad \theta\left(\left\lfloor \frac{u}{p} \right\rfloor, \left\lfloor \frac{v}{p} \right\rfloor\right) + x, \tag{8}$$

where $\theta(\cdot,\cdot)$ is the cluster-offset, defined in (7), and x is the *edge-offset* $x \in [0, 2k)$, if $(i \ne j)$, and $x \in [0,k)$, if $(i = j)$, which is computed at the time of edge insertion (by Algorithm 1, as described in Sec. 4 below).

The total number of spine-layer p-port switches to connect n leaf-layer switches in a bounded-degree ($\le k$) network used by OpticNet is:

$$\begin{aligned} n' \quad &= \quad |SW| = |SW^{\cup}| + |SW^{\times}| \\ &= \quad kC + 2k\binom{C}{2} = k \cdot C^2, \end{aligned} \tag{9}$$

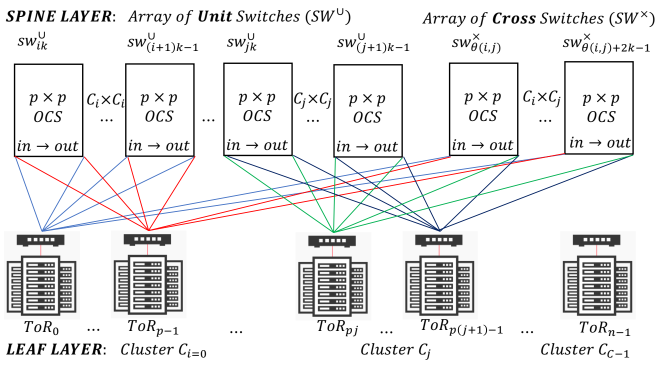

Fig. 1. OpticNet architecture: Leaf Layer: bounded-degree ($\le k$) network on n nodes; Spine Layer: arrays of unit and cross-type p-port switches $SW = SW^{\cup} \cup SW^{\times}, |SW^{\cup}| = kC, |SW^{\times}| = 2k\binom{C}{2}, C = \lceil n/p \rceil$.

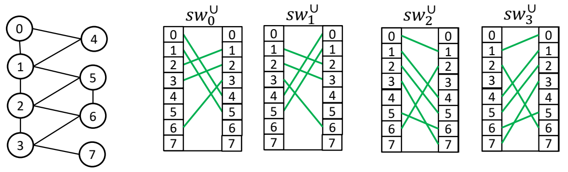

Fig. 2. OpticNet configuration example on a network with $k = 4$, $n = p = 8$, $C = 1$, $|SW^{\cup}| = kC = 4$ (no cross switches are needed).

where $\binom{C}{2}$ denotes the binomial coefficient, and $C = \lceil n/p \rceil$.[5]

Figure 1 illustrates how OpticNet implements a TMT architecture in a datacenter comprised of a bounded ($\le k$)-degree network on n nodes using aspine layer of constant-size p-port OCS switches. The set of n leaf-layer (ToR) switches is partitioned into C clusters of size p, and each $v \in C_i$ is connected to one input and one output port of k unit switches, and either one input or one output port of $2k$ cross switches, for each cluster $C_j, j \ne i$. The number of ports of each leaf-layer (static) switch is thus equal to $2k + 2k(C-1) = 2kC$. Note that, for each cluster-offset $\theta(i,j)$, there are k switches with in$\rightarrow$out port-sets $\mathcal{P}(sw_{idx}) = C_i \times C_j$ (i.e., $C_i \rightarrow C_j$) and another k switches with port-sets $\mathcal{P}(sw_{idx}) = C_j \times C_i$ (i.e., $C_j \rightarrow C_i$), $0 \le i \le j < C$. Figure 2 illustrates a network comprised of eight nodes of degree at most four ($n = 8$, $k = 4$), connected using four eight-port ($|SW^{\cup}| = 4$, $p = 8$) unit switches. Note that no cross switches are needed in this case, since $C = \lceil n/p \rceil = 1$.

Binary Search Tree (BST) Networks: SAN algorithms based on splaying (e.g. [7], [36], [39]) typically assume that N_t is a BST. We can use the properties of a BST topology to reduce the total number of OCS switches (and simplify the edge insertion process). Optimization in a particular system will depend on the OCS hardware characteristics, such as reconfiguration latency and number of ports, as well as the available budget for OCS hardware acquisition. In sections 3.1 and 3.2, we present two possible optimizations of the OpticNet architecture, specifically for BST SANs.

5. In the special case $p < k$, we replace k by p in Eq. (9), since there are at most p nodes per cluster, and $(k-p-1)$ (or $(k-p)$) neighbors of each node in a unit (or cross) cluster will belong to other clusters, so there is no need for $> p$ copies of each switch.

3.1 AugPath-BST OpticNet

Let T be a BST on a set V of n nodes and $\mathcal{M}_t$ be the set of matching computed by OpticNet at time t. Let $u \in C_i \subseteq V$ and $v \in C_j \subseteq V, i < j$. Even though T has bounded degree at most $k = 3$, the subgraph induced on the vertices of any pair of distinct clusters C_i and C_j in $\mathcal{M}_t$ has degree ≤ 2, for the following reason. Since $i < j$, each vertex $u \in C_i$ has a smaller identifier than any vertex $v \in C_j$. This means that u can have at most one child and one parent in C_j at a given point in time. Therefore, it is possible to represent the edges in $C_i \times C_j$ with only four cross switches. So, instead of $3C + 6\binom{C}{2}$, as in Eq. (9), the total number of switches in the spine layer becomes:

$$|SW|_{BST}^{AugPath} = 3C + 4\binom{C}{2}. \tag{10}$$

This would be optimum in terms of the total number of spine-layer switches and incur significant gain in terms of adjustment cost, as we show in Section 6.

3.2 Mirrored-BST OpticNet

An alternative solution for BST networks is to partition the set of spine-layer switches into four subsets, according to two criteria: (1) downward (parent→child) × upward (child→parent), and (2) right ($u \rightarrow v, u < v$) × left ($u \rightarrow w, w < u$):

$$SW^{M\times} = SW_{DL}^{M\times} \cup SW_{DR}^{M\times} \cup SW_{UL}^{M\times} \cup SW_{UR}^{M\times}. \tag{11}$$

Consider a BST T on a set of nodes V, and any pair of nodes $(u, v) \in T$, $u \in C_i \subseteq V$, $v \in C_j \subseteq V$, $i < j$. If there is a directed edge between u and v in T, it will belong to exactly one switch in one of the four subsets defined in (11).

The same logic can be applied to the set $SW^{M\cup}$ of unit clusters (the case $(i = j)$), as long as we use four unit switches, instead of three, per cluster. This would result in a total of

$$|SW|_{BST}^{mirror} = 4C + 4\binom{C}{2} \tag{12}$$

switches (instead of $3C + 6\binom{C}{2}$, as in Eq. (9)).

Edge indexing: Mirrored-BST OpticNet simplifies the edge-indexing process. We can map every downward right-edge (DR) to edge-index $x_{DR} = 0$ and every downward left-edge (DL) to edge-index $x_{DL} = 1$. Analogously, we can map every upward left-edge (UL) to edge-index $x_{UL} = 2$ and every upward right-edge (UR) to edge-index $x_{UR} = 3$. The switch-index of any edge $(u, v) \in T$, $u \in C_i \subseteq V$, $v \in C_j \subseteq V, i < j$, can thus be computed directly as follows (the cluster-index $\theta(i, j)$ has been defined in (7)):

$$\begin{aligned} swt\_idx(u, v, p) \;\; = \;\; & \theta\,(i, j) \; + \\ & (u.isParent(v) \; ? \; 0 \;:\; 2) \; + \\ & (u < v \; ? \; 0 \;:\; 1). \end{aligned} \tag{13}$$

Figure 3 illustrates a Mirrored-BST network of eight nodes ($n = 8$, $k = 3$), connected via eight unit-type and four cross-type 4-port switches ($p = 4$, $C = 2$,$|SW^{\cup}| = 4C = 8$, $|SW^{\times}| = 4\binom{2}{2} = 4$). In this example, by applying Eq. (13) for edge $(0, 4)$, we have that $i = 0, j = 1, \theta(0, 1) = 8, x_{DR} = 0$, so $sw\_idx(0, 4, 4) = (8 + 0) = 8$. On the other hand, for edge $(4, 0)$, $x_{UR} = 3$, so $sw\_idx(4, 0, 4) = (8 + 3) = 11$.

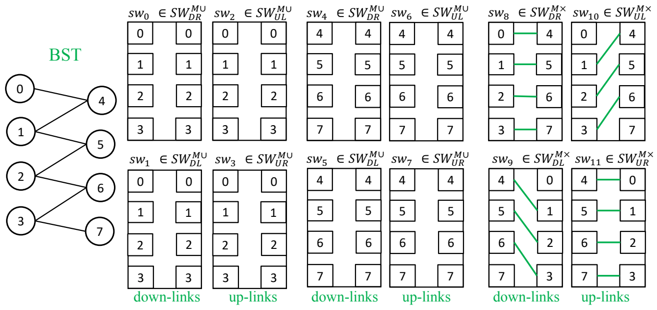
$sw_0 \in SW_{DR}^{M\cup}$
$sw_2 \in SW_{UL}^{M\cup}$
$sw_4 \in SW_{DR}^{M\cup}$
$sw_6 \in SW_{UL}^{M\cup}$
$sw_8 \in SW_{DR}^{M\times}$
$sw_{10} \in SW_{UL}^{M\times}$
BST
$sw_1 \in SW_{DL}^{M\cup}$
$sw_3 \in SW_{UR}^{M\cup}$
$sw_5 \in SW_{DL}^{M\cup}$
$sw_7 \in SW_{UR}^{M\cup}$
$sw_9 \in SW_{DL}^{M\times}$
$sw_{11} \in SW_{UR}^{M\times}$
down-links
up-links
down-links
up-links
down-links
up-links

Fig. 3. Mirrored-BST OpticNet example ($n = 8$, $p = 4$, $k = 3$, $C = 2$ $|SW^{\cup}| = 4C = 8, |SW^{\times}| = 4\binom{C}{2} = 4$).

4 Network Adjustments in OpticNet

We assume that, at each round t, OpticNet receives a set of edges $new\_edges \leftarrow N_{t+1} \setminus N_t$ to be inserted, and a set of edges $delete\_edges \leftarrow N_t \setminus N_{t+1}$ to be deleted, between two consecutive network topologies $N_t \in \mathcal{N}$ and $N_{t+1} \in \mathcal{N}$. We denote by $\mathcal{M}_t, \mathcal{M}_{t'}, \mathcal{M}_{t+1}$ the sets of matchings before, during, and after the network adjustment operation, respectively. Initially, we set $\mathcal{M}_{t'} \leftarrow \mathcal{M}_t$ and $\mathcal{M}_{t+1} \leftarrow \mathcal{M}_t$, and remove all edges in $delete\_edges$ from $\mathcal{M}_{t'}$. Then, each directed edge $(u, v) \in new\_edges$ is added to the new configuration, as described below.

Algorithm 1 describes the procedure of adding a *directed* edge $(u, v) \notin N_t$ to $\mathcal{M}_{t'}$[6] Firstly, from the subset of k unit or $2k$ cross switches $S_u^v = \{sw_z \in SW | \mathcal{P}(sw_z) = C_i \times C_j, u \in C_i, v \in C_j\}$, two switches, to which we refer as *opposite switch pair*, are selected: one switch (sw_x), where an input port (in_x^u) is free, and one switch (sw_y) in which the output port (out_y^v) is free (lines $1-6$). Note that it is possible that $sw_x = sw_y$, but at least one sw_x and one sw_y must exist. Otherwise, the addition of (u, v) would imply degree $> k$ for u or v in N_{t+1}.

The algorithm works around the idea of an augmenting path between the pair of *opposite* matchings $M(x, t') \in \mathcal{M}_{t'}$ and $M(y, t') \in \mathcal{M}_{t'}$, on switches sw_x and sw_y, respectively (lines $8-20$). The boolean variable $check\_out$ is initially set to $true$ (line 7) because we have to check for conflict (adjacent edge) on the output port out_x^v of switch sw_x. (It will be set to $false$ when the input port of some switch needs to be checked for a conflict). Note that, by choice of sw_x, node u does not have a neighbor connected to a port of sw_x; otherwise, the input port in_x^u would not have been free. So, it is sufficient to check the v node's neighbors (lines $09-14$). If there is a conflict, we remove the conflicting edge $(w, v) \in M(x, t')$ and set $check\_out$ to false. In the next iteration, we add (w, v) to the opposite matching $M(y, t')$ and (since $check\_out$ is false) check for conflict on the input port in_y^w of sw_y (lines $15-20$). Note that there can be only one conflicting port at a time (either on the output or on the input side of a switch). This process is repeated $\leq 2(p-1)$ times until there are no more conflicting edges in $M(x, t')$ or $M(y, t')$ (line 08).

Batch editing: Algorithm 2 adds an optimization to Algorithm 1 when multiple edges are inserted at once

6. Note that, if switches are simplex and edges are undirected, we will need two calls of Algorithm 1, so that both directed edges (u, v) and (v, u) are inserted into two, possibly distinct, matchings.

(e.g., a bidirectional edge, or a set of edges comprising one topological adjustment, such as a splay). Each edge $(u, v) \in new\_edges$ is inserted into an intermediate network state $\mathcal{M}_{t'}$ by making a call to Algorithm 1 (lines $3-5$). Then all edges in $\mathcal{M}_t \setminus \mathcal{M}_{t'}$ are removed from $\mathcal{M}_{t+1}$ (lines $6-7$), and all edges in $\mathcal{M}_{t'} \setminus \mathcal{M}_t$ are added directly to $\mathcal{M}_{t+1}$ (lines $8-9$). This way, redundant updates are avoided when several edges are inserted into the same switch.

Lazy adjustments: Even though batch-editing reduces the reconfiguration cost of simple operations, like splaying, an even more significant gain could potentially be achieved in *Lazy* SANs [40], in which topology reconfigurations are performed less often, or in so-called epochs. During one epoch, the physical network remains unmodified while adjustments are performed on a virtual copy of the network graph, and after each epoch, both network versions are synchronized.

Algorithm 1 AugPath(): single edge insertion

```
Require: $\mathcal{M}_{t'}$, $SW^{\cup}$, $SW^{\times}$, $k$, $(u,v) \notin N_t, u \in C_i, v \in C_j$.
1:  if $i = j$ then  %(unit switch)
2:      $S_u^v \leftarrow \{sw_z \in SW^{\cup} | \ \theta(i,j) \leq z < \theta(i,j) + k\}$
3:  else  %(cross switch)
4:      $S_u^v \leftarrow \{sw_z \in SW^{\times} | \ \theta(i,j) \leq z < \theta(i,j) + 2k\}$
5:  $sw_x \leftarrow sw_z \in S_u^v | (u,w) \notin M(z,t') \in \mathcal{M}_{t'}, \forall w \in C_j$
6:  $sw_y \leftarrow sw_z \in S_u^v | (w,v) \notin M(z,t') \in \mathcal{M}_{t'}, \forall w \in C_i$
7:  $check\_out \leftarrow true$
8:  while $(u,v) \notin (M(x,t') \cup M(y,t'))$ do
9:      if $check\_out$ then
10:         add $(u,v)$ to $M(x,t')$
11:         if $\exists w \in C_i \ / \ \{u\}, (w,v) \in M(x,t')$ then
12:             $check\_out \leftarrow false$
13:             remove $(w,v)$ from $M(x,t')$
14:             $(u,v) \leftarrow (w,v)$
15:     else
16:         add $(u,v)$ to $M(y,t')$
17:         if $\exists w \in C_j \ / \ \{v\}, (u,w) \in M(y,t')$ then
18:             $check\_out \leftarrow true$
19:             remove $(u,w)$ from $M(y,t')$
20:             $(u,v) \leftarrow (u,w)$
21: return $\mathcal{M}_{t'}$
```

Algorithm 2 BatchEdit(): set-of-edges insertion

```
Require: $\mathcal{M}_t$, $SW^{\cup}$, $SW^{\times}$, $k$, $new\_edges$.
1:  $\mathcal{M}_{t'} \leftarrow \mathcal{M}_t$
2:  $\mathcal{M}_{t+1} \leftarrow \mathcal{M}_t$
3:  for $(u,v) \in new\_edges$ do
4:      if $(u,v) \notin M(z,t'), \forall M(z,t') \in \mathcal{M}_{t'}$ then
5:          $\mathcal{M}_{t'} \leftarrow$ AugPath($\mathcal{M}_{t'}$, $SW^{\cup}$, $SW^{\times}$, $k$, $(u,v)$)
6:  for $(u,v) \in M(z,t) \in \mathcal{M}_t \mid (u,v) \notin M(z,t') \in \mathcal{M}_{t'}$ do
7:      remove $(u,v)$ from $M(z,t+1) \in \mathcal{M}_{t+1}$
8:  for $(u,v) \in M(z,t') \in \mathcal{M}_{t'} \mid (u,v) \notin M(z,t) \in \mathcal{M}_t$ do
9:      add $(u,v)$ to $M(z,t+1) \in \mathcal{M}_{t+1}$
10: return $\mathcal{M}_{t+1}$
```

BST optimizations: If we use the Mirrored-BST OpticNet (described in Sec. 3.2), switch indexing can be computed using Eq. (13), so there is no need to call Algorithm 1. However, it is important to note that inversions in parent→child relations (e.g., when a node becomes the parent of a former child due to a rotation, such as $zigzig$ or $zigzag$ [37]) might trigger additional link updates, and this overhead can outweigh the (p-factor) gain, in some networks, as we show in Section 6. If we use the AugPath-BST OpticNet (described in Sec. 3.1), the batch-editing (Algorithm 2) would no longer be needed, since there would be no conflicting paths between inserted edges (even though it could still be advantageous for lazy reconfigurations [40]).

4.1 OpticNet framework implementation

We implemented OpticNet as a simulation framework for SAN algorithms in the context of TMT architectures. In this framework, the decision-making and network adjustment specification of a SAN algorithm can be easily added or extended. We adopted an optimistic approach to solving concurrency conflicts, like some of the earlier work on concurrent SANs [36], [39]. To avoid deadlocks and starvation, messages that have been in the network for longer receive higher priority than messages issued more recently.

Centralized control node: All communication between the leaf and the spine layers is implemented via a *centralized controller node* that maintains a current global view of the network topology and coordinates port-mapping decisions among ToR and OCS switches. Moreover, the controller node handles prioritization rules among nodes participating in each (routing and forwarding) operation to ensure they don't join in conflicting operations.

Rounds and time-slots: The simulation is divided into synchronous rounds, and a message travels up to one hop per round. Each round is divided into 4 time-slots, as follows: `1: nodeInformStep`: Leaf-layer (ToR) switches (nodes) inform the controller node if they have messages to send; `2: controllerStep`: The controller node receives the network messages and computes the current round adjustment operations, informing all leaf and spine-layer switches which edges they must alter and the network nodes about their new neighbors (e.g., children or parent), and grants permission to selected nodes to forward their messages; `3: nodeRoutingStep`: The nodes that received permission to forward their messages do so; `4: logRoundResults`: Network nodes receive their messages, inform the controller node if they are their message's final destination, and the controller node logs all relevant information about that round.

Multi-round adjustments: In the case of BST-based SANs, most adjustments take 2 rounds, and some (e.g. a *zig-zag* splay) take 3 rounds. To increase concurrency, OpticNet handles conflicts around such adjustments differently from some previous work [27]. Nodes participating in an adjustment operation are not locked for more than one round. Messages related to unfinished adjustments receive maximum priority to prevent multi-round adjustments from being interrupted by messages with higher priority.

OpticNet-SAN integration: The integration of OpticNet with SAN algorithms involves three steps. First, create a new controller node class that inherits from the `NetworkController` class, add a constructor for it with the SAN algorithm specifications, and a call to the parent class constructor. Then, all decision-making and attribute updating specific to the SAN algorithm (e.g., rank computation, path weight update, or random number generation, in the case of a CBNet [39] implementation) is implemented as an override of the adjustment operations or message handling. Finally, creating a `CustomGlobal` class responsible for calling the network constructor, firing messages into the network, defining the stop condition, and any simulation configuration specifications is necessary.

Source code: The source code for the OpticNet implementation is available at [41]. This implementation assumes $\mathcal{N}$ to be the family of all BSTs and that switches are simplex, with each switch having a reversed (mirrored) copy added to the spine layer. Below, we describe examples of SAN algorithms implemented over OpticNet.

5 Analysis

In this section we analyze the correctness and optimality of OpticNet in the SMM. First we prove that Algorithm 1 terminates and correctly inserts new edges into the network.

Lemma 1. *Let $\mathcal{N}$ be the family of bounded-degree ($\leq k$) graphs on a set V of n nodes. Let $SW = SW^{\cup} \cup SW^{\times}$ be the set of spine-layer p-port (OCS) switches, $\mathcal{M}_t = N_t \in \mathcal{N}$ be the set of matchings computed by OpticNet at time t, and let $(u,v) \notin N_t$ be a directed edge to be inserted into the network. If $N_t \cup \{(u,v)\} \in \mathcal{N}$ then Algorithm 1 terminates in at most $2(p-1)$ iterations and returns a new set of matchings $\mathcal{M}_{t'} = N_t \cup \{(u,v)\} \in \mathcal{N}$, s.t. $\exists sw_x \in SW \mid (u,v) \in M(x,t'), M(x,t') \in \mathcal{M}_{t'}$.*

Proof. OpticNet assigns either $2k$ *cross* (if $i \neq j$) or k *unit* (if $i = j$) switches $sw_z \in SW \mid \mathcal{P}(sw_z) = C_i \times C_j, u \in C_i \subseteq V, v \in C_j \subseteq V, i \leq j$. We refer to this set as S_v^u. Note that, in both cases (*cross* or *unit* switch type), there are k switches in S_v^u on which u is connected to an input port (in_z^u), and k switches on which v is connected to an output port (out_z^v).

Since the out-degree of u and the in-degree of v must be $< k$ in N_t, there exists at least one switch $sw_x \in S_v^u$ (mapped to the matching $M(x,t) \in \mathcal{M}_t$), such that the input port in_x^u is free, and at least one $sw_y \in S_v^u$ (mapped to $M(y,t) \in \mathcal{M}_t$), such that the output port out_y^v is free.

In the case ($x = y$), we have that $M(x,t') \leftarrow M(x,t) \cup \{(u,v)\}$, and the proof is done. In the case ($x \neq y$), let $M = M(x,t) \cup M(y,t) \cup \{(u,v)\}$. Note that $|M| \leq 2(p-1)+1$, and M is comprised of edges that do not conflict at any ports on sw_x or sw_y, except for the new edge (u,v), which might conflict with at most one edge on sw_x.

Algorithm 1 (lines $7-20$) partitions M into two disjoint matchings $M(x,t')$ and $M(y,t')$ of size $\leq p$, using the idea of an augmenting path in M. Starting with the port-pair (edge) (in_x^u, out_x^v), it checks if there is a conflicting edge at port out_x^v on sw_x. If yes, let's call this edge (in_x^w, out_x^v), this edge is moved from sw_x to sw_y. Let's call this new edge (in_y^w, out_y^v). Note that, by choice of sw_y, the output port out_y^v has to be free. So the algorithm proceeds by checking for a conflict only at the input port in_y^w on switch sw_y. If there is no conflict on that port, the algorithm terminates and assigns all edges in M that remained on (or were moved to) sw_x to $M(x,t')$ and those on sw_y to $M(y,t')$.

We claim that Algorithm 1 does not repeat conflicting ports and, therefore, terminates after $\leq 2(p-1)$ iterations. For the sake of contradiction, suppose some conflicting port is repeated. W.l.g., let's assume it is an output port on sw_x and call it out_x^w. This means that M contains the pair of edges (e_1, e_2), $e_1 = (in_x^{w_1}, out_x^w)$, $e_2 = (in_x^{w_2}, out_x^w)$, $w_1 \neq w_2$, (because out_x^w has been processed in a previous iteration), and a third edge, which is either of type $e_3 = (in_x^{w_3}, out_x^w)$, $w_3 \neq w_2 \neq w_1$, or of the form $e_4 = (in_y^{w_3}, out_y^w)$ (i.e., it has been moved from sw_y to sw_x). In the first case, we have a contradiction, since there can be at most one conflicting pair of edges on sw_x (one of them being the new edge (u,v)). In the second case, because e_4 must have been moved from sw_y, there must be a conflicting (adjacent) edge $e_5 = (in_y^{w_4}, out_y^w)$ on sw_y, which contradicts the fact that $M(y,t)$ is a matching. This concludes the proof. □

We now prove that OpticNet can represent any bounded-degree ($\leq k$) network with a total of $|SW| = kC + 2k\binom{C}{2}$ spine-layer switches, i.e., every edge is assigned to a matching on some switch.

Theorem 1. *Let $\mathcal{N}$ be the family of bounded-degree ($\leq k$) graphs on a set V of n nodes. Let SW be a set of spine-layer p-port switches and $\mathcal{M}_t = N_t \in \mathcal{N}$ be the set of matchings computed by OpticNet at time t. We have that $\forall (u,v) \in V \times V$, if $(u,v) \in N_t$, then $\exists sw_x \in SW \mid (u,v) \in M(x,t), M(x,t) \in \mathcal{M}_t$.*

Proof. W.l.g, we assume that edges are directed, i.e., switches are simplex. The proof is by induction on the number of edges.

Base case: If N_t has no edges, the claim is vacuously true since $(u,v) \notin N_t, \forall (u,v) \in V \times V$.

I.H.: $\forall (u',v') \in V \times V$, if $(u',v') \in N_t' = N_t \setminus (u,v)$ then $\exists sw_{x'} \in SW \mid (u',v') \in M(x',t),\ M(x',t) \in \mathcal{M}_t' = N_t'$.

By I.H., every edge in N_t' is assigned to some matching in $\mathcal{M}_t'$. Consider an edge $(u,v) \notin N_t'$ such that $N_t = N_t' \cup \{(u,v)\} \in \mathcal{N}$, i.e., the insertion of the new edge preserves degree $\leq k$ of the underlying graph. By Lemma 1, Algorithm 1 returns a set of matchings $\mathcal{M}_t = N_t$, such that $\exists sw_x \in SW \mid (u,v) \in M(x,t), M(x,t) \in \mathcal{M}_t$. □

The correctness of AugPath-BST OpticNet using $|SW|_{BST}^{AugPath} = 3C + 4\binom{C}{2}$ (Eq. 10) switches follows directly from Theorem 1 ($k = 3$) and the fact that any subgraph induced on the vertices of two distinct clusters in a BST has degree ≤ 2. Therefore, four (rather than six) cross-switches per cluster-pair are sufficient to realize any BST network.

The following theorem proves the correctness of the Mirrored-BST OpticNet, which uses a total of $|SW|_{BST}^{mirror} = 4C + 4\binom{C}{2}$ switches (Eq. 12), as described in Section 3.2.

Theorem 2. *Let T_t be any BST on a set V of n nodes at time t, let $\mathcal{M}_t$ be the set of matchings and $SW^M = SW_{DL}^M \cup SW_{DR}^M \cup SW_{UL}^M \cup SW_{UR}^M$ be the set of simplex switches computed by Mirrored-BST OpticNet at time t. Then $\forall (u,v) \in T_t : \exists sw_x \in SW^M | (u,v) \in M(x,t) \in \mathcal{M}_t$.*

Proof. Let $u \in C_i \subseteq V$ and $v \in C_j \subseteq V$. Mirrored-BST OpticNet assigns either four *cross* (if $i \neq j$) or four *unit* (if $i = j$) switches $sw_z \in SW^M \mid \mathcal{P}(sw_z) = C_i \times C_j$. We refer to this set as S_v^u and denote by $M(z,t) \in \mathcal{M}_t$ the matching on a switch $sw_z \in S_v^u$.

W.l.g, let's assume that u is the parent of v in T_t, and $u > v$, i.e., v is a left-child of u, and let $sw_x = S_v^u \cap SW_{DL}^M$. Suppose v's output port (out_x^v) is occupied by another edge on sw_x. This would imply that $\exists(w, v) \in M(x,t) | w \neq u$, which is a contradiction since v can have at most one parent in T_t. Suppose on the other hand that u's input port on sw_x (in_x^u) is occupied by a different edge. This would imply that $\exists(u, w) \in M(x,t) | w \neq v$, again a contradiction since u can have at most one left child in T_t. The same logic applies to the remaining three cases ($sw_x \in \{SW_{DR}^M, SW_{UL}^M, SW_{UR}^M\}$). □

We proceed by showing that the number of spine-layer switches is optimal.

Theorem 3. *Let $\mathcal{N}$ be the family of bounded-degree ($\leq k$) graphs on a set V of n nodes, and OPT we the minimum-size set of spine-layer switches with p in$\rightarrow$out ports needed to connect n leaf-layer switches via a TMT two-layer architecture in the SMM. Let SW be a set of spine simplex switches computed by OpticNet and $C = \left\lceil \frac{n}{p} \right\rceil$, then*

$$|OPT| \geq |SW| = k \cdot C^2.$$

Proof. For simplicity's sake, we assume that loop edges are possible $(v, v), \forall v \in V$ in some network topologies $N_t \in \mathcal{N}$.

Let $\mathcal{D}(V) = \sum_{v \in V} \mathcal{D}(v)$ be the total number of input and output ports of n leaf-layer (ToR) switches, and let $|OPT|$ be the number of spine-layer (optical) switches with p input and p output ports in any feasible solution OPT for the SMM-PMP.

Since each input (output) port allows a node $u \in V$ to connect to at most one of p output (input) ports ($v \in V$) through some switch $sw_x \in OPT$, each $u \in V$ needs $\geq 2k \cdot C$ links to be able to connect to any k neighbors in N_t simultaneously (in both directions), which means that $\mathcal{D}(V) \geq n \cdot 2k \cdot C$. Furthermore, since each switch $sw_x \in OPT$ has $2p$ ports, we have that $|OPT| \geq \mathcal{D}(V)/2p = kC^2$. □

Finally, we show that any SAN algorithm runs on top of OpticNet, preserving the total cost guarantees.

Theorem 4. *Let $\mathcal{N}$ be the family of bounded-degree ($\leq k$) graphs on a set of n nodes. Consider any initial $N_0 \in \mathcal{N}$ (and a BST T_0), a sequence of m messages σ and a SAN algorithm $\mathcal{A}$. Let SW be the set of spine-layer p-port switches and $\mathcal{M}_t = \cup_{sw_x \in SW} M(x,t)$ be the set of matchings computed by OpticNet at time t, such that $\mathcal{M}_t = N_t \in \mathcal{N}$. The total service, edge and switch adjustment, and work costs incurred by $\mathcal{A}$ on top of OpticNet (and Mirrored-BST $OpticNet^M$) to deliver all messages in σ are:*

$$\begin{aligned}
\mathit{Srv}(OpticNet(\mathcal{A}), N_0, \sigma) &= \mathit{Srv}(\mathcal{A}, N_0, \sigma), \\
\mathit{LiAdj}(OpticNet(\mathcal{A}), N_0, \sigma) &\leq 2p \cdot \mathit{LiAdj}(\mathcal{A}, N_0, \sigma), \\
\mathit{LiCost}(OpticNet(\mathcal{A}), N_0, \sigma) &= O(\mathit{LiCost}(\mathcal{A}, N_0, \sigma)), \\
\mathit{SwAdj}(OpticNet(\mathcal{A}), N_0, \sigma) &\leq 2 \cdot \mathit{LiAdj}(\mathcal{A}, N_0, \sigma), \\
\mathit{SwCost}(OpticNet(\mathcal{A}), N_0, \sigma) &= O(\mathit{SwCost}(\mathcal{A}, N_0, \sigma)), \\
\mathit{LiAdj}(OpticNet^M(\mathcal{A}), T_0, \sigma) &= \mathit{LiAdj}(\mathcal{A}, T_0, \sigma), \\
\mathit{SwAdj}(OpticNet^M(\mathcal{A}), T_0, \sigma) &= \mathit{LiAdj}(\mathcal{A}, T_0, \sigma).
\end{aligned}$$

Proof. All bounds follow directly from Theorems 1 and 2. □

6 Evaluation

In this section, we present our experimental results. Firstly, we implemented three SAN algorithms on top of Mirrored-BST OpticNet:

- SN [42]: SplayNet [7], a centralized and sequential generalization of splay trees [37];
- DSN [42]: Semi-DiSplayNet [39], a distributed and concurrent version of SplayNet with semi-splays that aim to improve message routing;
- CBN [43]: CBNet [27], a distributed and concurrent generalization of CBTrees [38].

Moreover, we evaluated the BST-specific optimizations proposed in Sections 3.1 and 3.2, implementing another four combinations:

- $OpticNet^{AP}(ODSN)$ [44]: Original DiSplayNet [7] in combination with the AugPath-BST OpticNet, as defined in Sec. 3.1.
- $OpticNet(ODSN)$ [44]: Original DiSplayNet [36] in combination with the Mirrored-BST OpticNet, as defined in Sec. 3.2.
- $OpticNet^{AP}(DSN)$ [42]: Semi-DiSplayNet [39] in combination with the AugPath-BST OpticNet.
- $OpticNet(DSN)$ [42]: Semi-DiSplayNet in combination with the Mirrored-BST OpticNet.

We used the following metrics, defined in Section 2, to evaluate the simulation results for an algorithm $\mathcal{A}$:

- *Service cost*: $\text{Srv}(\mathcal{A}, N_0, \sigma)$ (Def. 1).
- *Link-adjustment cost*: : $\text{LiAdj}(\mathcal{A}, N_0, \sigma)$ (Def. 2).
- *Switch-adjustment cost*: $\text{SwAdj}(\mathcal{A}, N_0, \sigma)$ (Def. 3).
- *Total link cost (work)*: $\text{LiCost}(\mathcal{A}, N_0, \sigma)$ (Def. 4).
- *Total switch cost (work)*: $\text{SwCost}(\mathcal{A}, N_0, \sigma)$ (Def. 5).
- *Workload balance*: the percentage of time (simulation rounds) a node, port, or switch has been active (forwarding a packet) in a routing or adjustment operation, by the end of each simulation.

6.1 Workload traces

To measure the locality of reference present in a workload, we use the definition of *trace complexity* [21], which leverages only randomization and data compression operations. We measure the amount of locality present in a workload based on the entropy of the communication sequence. Entropy is related to the amount of information or the ability to compress the data. Intuitively, workloads with a low locality structure tend to have random sequences of communication pairs and, consequently, a low data compression rate. High-locality sequences tend to have a specific structure between requests that allows for better compression. In particular, we distinguish temporal and spatial components.

We used the following workload traces in our experiments, grouped according to their locality of reference:

7. In this early version of DiSplayNet [36], only bottom-up splaying was implemented, in contrast to bottom-up and top-down semi-splaying, adopted in later versions [39].

High spatial, low temporal: **ProjecToR** [45] describes the communication probability distribution among $8,367$ pairs of nodes in a network of $n = 128$ nodes (top of racks), randomly selected from 2 production groups, executed between map-reduce operations, index builders, database and storage systems. We sample a sequence of $m = 10,000$ independent orders and identically distributed ($i.i.d.$) in time by the given communication matrix and repeat each experiment 30 times.

High temporal, low spatial: **Bursty** was generated artificially using the method from [21] ($m = 10,000$, $n = 128$).

Low locality: **Facebook** [46] trace consists of real *Fbflow* packets collected from the production clusters of Facebook. The per-packet sampling is uniformly distributed with a rate of 1:30000, flow samples are aggregated every minute, and node IPs are anonymized. We mapped the anonymized IPs to a consecutive value range starting at 0. This resulted in a sequence of $1,000,000$ requests, originating in a 24-hour time window in a network comprising $n = 159$ nodes.

To space the requests in time and make the timestamp sequences more realistic, we used a Poisson distribution with $\lambda = 0.25$ to determine the time of the entrance of each message in the network.

6.2 Results

Work cost: In Figure 4 we analyze the total work, partitioned into routing cost (Srv$(\mathcal{A}, N_0, \sigma)$) and total link adjustment cost (LiAdj$(\mathcal{A}, N_0, \sigma)$), as defined in (1) and (4), resp. We observe that CBNet performs almost no adjustments, compared to SplayNet and DiSplayNet, and has a higher routing (service) cost only in the Bursty workload. This is due to its high temporal locality, which gives an advantage to more aggressive reconfiguration algorithms, like SplayNet and DiSplayNet. Note that these metrics are not influenced by the OpticNet framework but only by each SAN algorithm.

Workload distribution: In Figures 5 and 6 we present the CDF (Cumulative Distribution Function) of the percentage of active rounds and active ports per switch for all workloads, respectively. In all plots, the smallest switch size was used ($p = 8$) for a network with $n \in \{128, 144\}$ nodes. Analyzing the results in Figure 5, we observe that at least 50% of switches remain idle close to 100% of the time, and almost 100% of switches are active for less than 0.3% of simulation rounds, in all scenarios. The scenario that presented the most balanced activity distribution over time was DiSplayNet, the most adjustment-intensive among the SAN algorithms, in the Bursty workload, the one with the highest temporal locality. Figure 6 shows the percentage of active ports per switch that have been activated in at least one round. The results are similar to that of switch activation over time. We observe that CBNet presented the highest port idleness for all workloads, which can be explained by the fact that, among the simulated SAN algorithms, it performs the lowest number of adjustments in general.

These first experiments revealed an extremely high rates of idleness when the system is comprised of a large number of very small switches ($p = 8$). Next, we investigate the impact of the size of the switch (in number of ports) on the distribution of switch activity over time.

Variable switch size: In Figure 7 we analyze the impact of increasing switch size on the workload distribution for the ProjecToR dataset ($n = 128, p \in \{8, 32, 128\}$). We observe that by increasing the number of ports, the overall idleness of the system decreases sharply. For CBNet, the percentage of switches that remain inactive 100% of the time drops from over 80% to 40%, for $p = 8$ and $p = 32$, resp. For SplayNet and DiSplayNet, this decrease is from 60% to 20%. For $p = 128$, idleness drops to 0% for all algorithms, which is expected in a network with $p = n$. For $p = 32$, a more realistic scenario, we observe that DiSplayNet utilizes at least 50% of switches for at least 0.1% of simulation rounds, 25% of switches for at least 0.2% of rounds, and approximately 10% of switches for at least 0.3% of time. Both CBNet and SplayNet activate at least 25% of switches for more than 0.1% of rounds. Overall, we observe a significant improvement in workload distribution as the number of ports per switch increases.

AugPath-BST: In Figure 8 we analyze the total link adjustment cost ($LiCost$, Def. (4)) of AugPath-BST OpticNet. As a baseline, we used the Mirrored-BST implementation of OpticNet for BST networks (the one used in the previous experiments). Moreover, we simulated two versions of DiSplayNet, with (DSN [39]) and without (ODSN [36]) top-down semi-splaying. Note that in the former (DSN), the number of link updates per rotation is lower than in the latter (ODSN), which is an earlier implementation version of DiSplayNet, at the expense of slightly higher service cost.

We observe that in all plots, AugPath-BST has a significantly lower (up to 40%) total (link) adjustment cost than its Mirrored-BST counterpart. This behavior can be explained as follows. The parent$\rightarrow$child relation between nodes participating in some splaying rotations is inverted, i.e., a node $u \in T_t$ that was a parent of $v \in T_t$ might become a child of v in T_{t+1}, after a $zigzig$ or a $zigzag$ [37]. In Mirrored-BST, this might trigger additional link updates for the (u, v) edge because switches in SW^D store only downwards (parent$\rightarrow$child) edges, while switches in SW^U store only upwards (child$\rightarrow$parent) edges. The same applies to left$\rightarrow$right relation inversions. For example, to perform a $zigzig$ and a $zigzag$ rotation, Mirrored-BST triggers 10 link-updates, while AugPath-BST does only 6 and 8, respectively.

To sum up, AugPath-BST has the advantage of using fewer unit switches (3 instead of 4 per cluster) and triggering significantly fewer link and switch updates when executing certain network adjustments. On the other hand, Mirrored-BST simplifies edge-indexing and edge insertion, while causing overhead in adjustment cost, which might be mitigated in scenarios with higher p-values or link densities.

Final remarks: Overall, our experiments showed that when OpticNet is built using switches of limited size ($p \ll n$), the rates of idleness in the system can be alarmingly high. At the same time, we observed significantly better workload distributions as the number of ports per switch increased. The practical implication of these results is that the cost and availability of OCS hardware will be crucial when it comes to the deployment of arbitrary (bounded-degree) reconfigurable datacenter networks. This reinforces the demand and motivation for advances in OCS technology.

7 Related Work

Reconfigurable network topologies can be grouped in two basic types: demand-oblivious, e.g. [1], [11], [12], [19], [47]

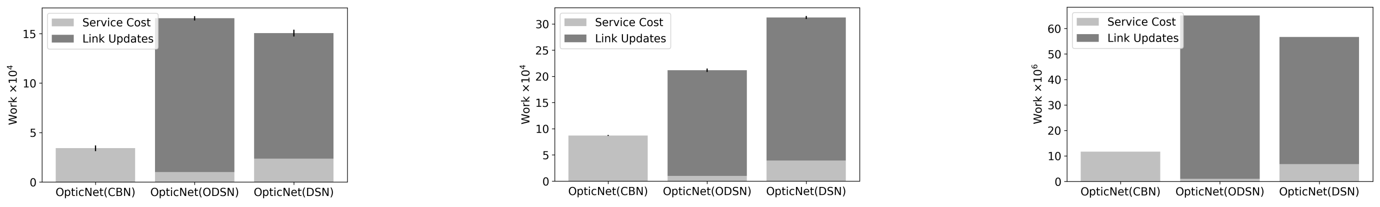

Fig. 4. Total work: $m = 10^4$: (a) ProjecToR, (b) Bursty; $m = 10^6$: (c) Facebook.

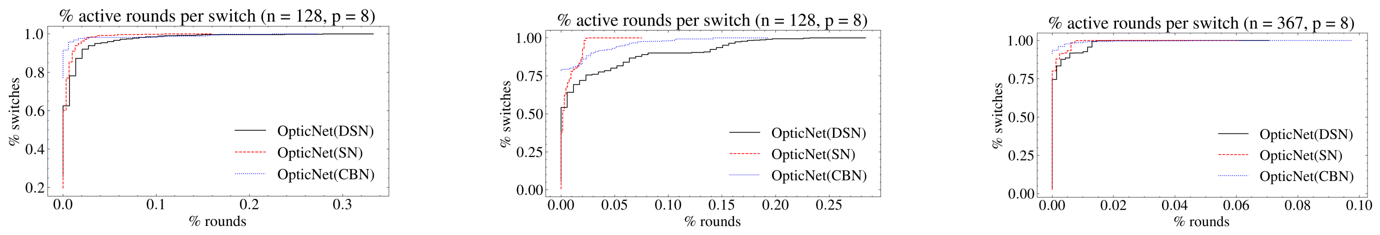

Fig. 5. Workload distribution. CDF Active rounds % per switch (fixed switch size $p = 8$): (a) ProjecToR, (b) Bursty, (c) Facebook.

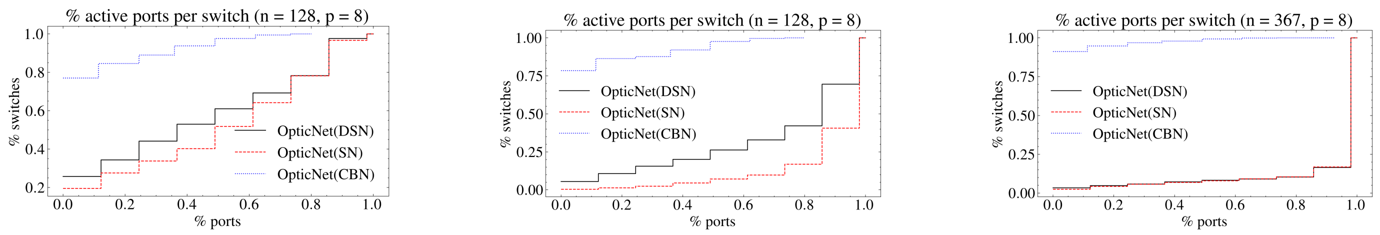

Fig. 6. Workload distribution. CDF Active ports % per switch (fixed switch size $p = 8$): (a) ProjecToR, (b) Bursty, (c) Facebook.

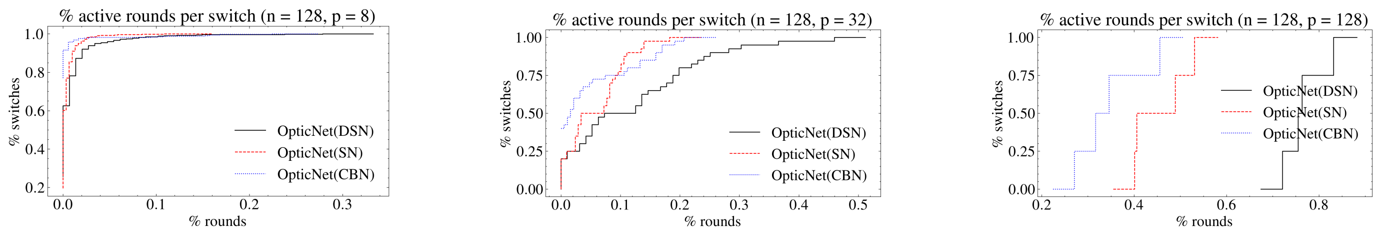

Fig. 7. Workload distribution. ProjecToR: CDF Active rounds % per switch (variable switch size $p \in \{8, 32, 128\}$).

and demand-aware, e.g. [6], [7], [8], [25], [34], [36], [40], [48], [49], [50], [51], [52], or a combination of both, e.g., Cerberus [32]. Most existing demand-aware architectures rely on an estimation of traffic matrices [8], [23], [50], [51], [53], [54] which can limit the granularity and reactivity of the network, but there are also more fine-grained approaches such as [27], [28], [34], [48].

The majority of optical circuit switching for datacenter networks with on-demand reconfigurations (RDCNs) have been based on full-crossbar 3D-MEMS OCS [24], Wavelength Division Multiplexing [23] and, more recently, integrated soliton microcombs and tunable lasers [1], [22]. Several prototypes based on OCS have been built, and their advantages demonstrated, e.g., the Helios prototype [8] uses a commercially available Glimmerglass OCS with 64 ports, while OSA [25] provides a test-bed that consists of a Polatis Series 1000 32-port OCS. To operate such RDCNs efficiently, the self-adjusting network topology has to be mapped to a set of OCS dynamically in real-time. There are also large-scale deployments like the Jupiter data center network by Google [13], [29], [30].

While promising performance results have been demonstrated with various prototypes of demand-aware reconfigurable networks, today, it is often challenging to experiment with these technologies, as they are usually based on custom-built prototypes and rely on tailored hardware and software which is not publicly available. One example of a framework that supports experimentation and reproducibility is ExReC [55]. It uses off-the-shelf hardware (Polatis Series 6000n 32 × 32 OCS [35]) for evaluating different hybrid reconfigurable topologies and applications.

The closest work to ours is [33], where online strategies are proposed to adapt SAN algorithms for the matching model [31], based on spaced out (less frequent) network adjustments using existing (sequential) SAN algorithms, such as SplayNets [7] and ReNets [34].

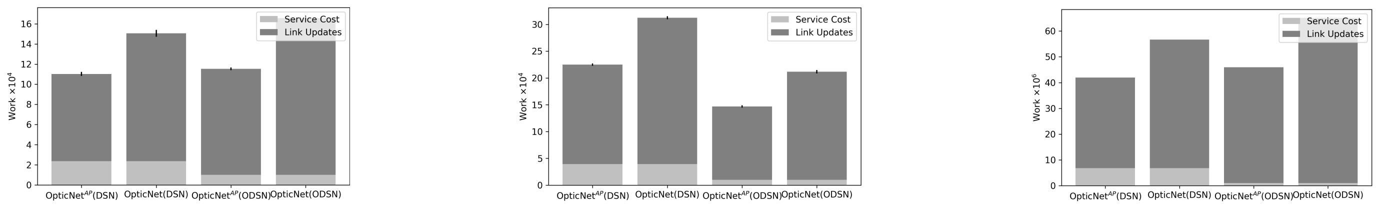

Fig. 8. AugPath-BST OpticNet: Link-adjustment total cost (switch size $p = 8$): (a) ProjecTor, (b) Bursty, (c) Facebook.

8 Conclusion

Even though emerging hardware reconfiguration technologies introduce an additional degree of freedom to the datacenter network design problem, they still face challenging system and algorithm design problems. Due to additional control logic and several milliseconds latency of the state-of-the-art demand-aware optical switches, their cost can only be amortized for large flows over longer terms.

In this work we presented SMM—a scalable matching model—and OpticNet—a simulation framework—for designing reconfigurable datacenter networks, inspired by the emerging OCS technologies. SMM adds flexibility to previous models by realizing any bounded-degree topology using (optical) switches of constant size. Our experimental results elucidated some important aspects of this setting, such as the trade-off between the size of the system (number of switches) and the size of each switch (number of ports).

A promising future direction would be to investigate SMM and OpticNet in combination with Lazy SANs, such as LazySplayNets and LazyReNets [40]. One of the benefits of these algorithms is the possibility to bind together the total number of (link or switch) adjustments and the actual (OCS hardware-dependent) cost to reconfigure an individual optical switch. To achieve that, one would have to carefully address the real-time processing and memory requirements for the virtual network state maintenance and the synchronization procedures.

Finally, we believe that, even though our work is still theoretical in nature, it nevertheless constitutes an interesting step toward practical self-adjusting networks.

References

[1] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen *et al.*, "Sirius: A flat datacenter network with nanosecond optical switching," in *Proc. ACM SIGCOMM*, 2020, pp. 782–797.

[2] H. Liu, R. Urata, K. Yasumura, X. Zhou, R. Bannon, J. Berger, P. Dashti, N. Jouppi, C. Lam, S. Li, E. Mao, D. Nelson, G. Papen, M. Tariq, and A. Vahdat, "Lightwave Fabrics: At-Scale Optical Circuit Switching for Datacenter and Machine Learning Systems," in *Proceedings of the ACM SIGCOMM 2023 Conference*, ser. ACM SIGCOMM '23, 2023, p. 499–515.

[3] J. Zerwas, C. Gyorgyi, A. Blenk, S. Schmid, and C. Avin, "Duo: A High-Throughput Reconfigurable Datacenter Network Using Local Routing and Control," in *ACM SIGMETRICS*, 2023.

[4] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2009.

[5] C. Avin and S. Schmid, "Toward demand-aware networking: A theory for self-adjusting networks," in *ACM SIGCOMM Computer Communication Review (CCR)*, 2018.

[6] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "ProjecToR: Agile reconfigurable data center interconnect," in *Proc. ACM SIGCOMM*. New York, NY, USA: ACM, 2016, pp. 216–229.

[7] S. Schmid, C. Avin, C. Scheideler, M. Borokhovich, B. Haeupler, and Z. Lotker, "Splaynet: Towards locally self-adjusting networks," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1421–1433, 2016.

[8] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 339–350, 2010.

[9] C. Avin, K. Mondal, and S. Schmid, "Demand-aware network designs of bounded degree," in *Proc. International Symposium on Distributed Computing (DISC)*, 2017.

[10] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: A reconfigurable wireless data center fabric using free-space optics," in *ACM SIGCOMM Computer Communication Review*, vol. 44. ACM, 2014, pp. 319–330.

[11] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, "Expanding across time to deliver bandwidth efficiency and low latency," *arXiv preprint arXiv:1903.12307*, 2019.

[12] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, "Rotornet: A scalable, low-complexity, optical datacenter network," in *Proc. ACM SIGCOMM*, 2017, pp. 267–280.

[13] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, vol. 45, no. 4, pp. 183–197, 2015.

[14] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A Fault-tolerant Engineered Network," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'13, 2013, pp. 399–412.

[15] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a high performance, server-centric network architecture for modular data centers," *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, vol. 39, no. 4, pp. 63–74, 2009.

[16] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: a high performance network structure for modular data center interconnection," in *Proc. ACM International Conference on Emerging Networking Experiments and Technologies (CONEXT)*, 2009, pp. 25–36.

[17] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *Proc. ACM SIGCOMM*, 2017, pp. 281–294.

[18] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers, randomly," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 12, 2012, pp. 17–17.

[19] V. Addanki, C. Avin, and S. Schmid, "Mars: Near-Optimal Throughput with Shallow Buffers in Reconfigurable Datacenter Networks," in *ACM SIGMETRICS*, 2023.

[20] K. Hanauer, M. Henzinger, L. Ost, and S. Schmid, "Dynamic Demand-Aware Link Scheduling for Reconfigurable Datacenters," in *IEEE INFOCOM*, 2023.

[21] C. Avin, M. Ghobadi, C. Griner, and S. Schmid, "On the complexity of traffic traces and implications," in *Proc. ACM SIGMETRICS*, 2020.

[22] A. S. Raja, S. Lange, M. Karpov, K. Shi, X. Fu, R. Behrendt, D. Cletheroe, A. Lukashchuk, I. Haller, F. Karinou, B. Thomsen,

K. Jozwik, J. Liu, P. Costa, T. J. Kippenberg, and H. Ballani, "Ultrafast optical circuit switching for data centers using integrated soliton microcombs," *Nature Communications*, vol. 12, no. 1, 2021.

[23] L. Chen, K. Chen, Z. Zhu, M. Yu, G. Porter, C. Qiao, and S. Zhong, "Enabling wide-spread communications on optical fabric with megaswitch," ser. NSDI'17, 2017, p. 577–593.

[24] J. Zerwas, W. Kellerer, and A. Blenk, "What you need to know about optical circuit reconfigurations in datacenter networks," in *The 33nd International Teletraffic Congress (ITC 33)*, 2021.

[25] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 2, pp. 498–511, 2014.

[26] V. Addanki, M. Pacut, A. Pourdamghani, G. Retvari, S. Schmid, and J. Vanerio, "Self-Adjusting Partially Ordered Lists," in *IEEE INFOCOM*, 2023.

[27] O. A. de O. Souza, O. Goussevskaia, and S. Schmid, "CBNet: Demand-aware tree topologies for Reconfigurable Datacenter Networks," *Computer Networks*, vol. 213, p. 109090, 2022.

[28] B. Peres, O. A. d. O. Souza, O. Goussevskaya, C. Avin, and S. Schmid, "Distributed self-adjusting tree networks," *IEEE Transactions on Cloud Computing*, 2021.

[29] A. D. Ferguson, S. Gribble, C.-Y. Hong, C. Killian, W. Mohsin, H. Muehe, J. Ong, L. Poutievski, A. Singh, L. Vicisano, R. Alimi, S. S. Chen, M. Conley, S. Mandal, K. Nagaraj, K. N. Bollineni, A. Sabaa, S. Zhang, M. Zhu, and A. Vahdat, "Orion: Google's Software-Defined Networking Control Plane," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 2021, pp. 83–98.

[30] R. Urata, H. Liu, K. Yasumura, E. Mao, J. Berger, X. Zhou, C. Lam, R. Bannon, D. Hutchinson, D. Nelson, L. Poutievski, A. Singh, J. Ong, and A. Vahdat, "Mission Apollo: Landing Optical Circuit Switching at Datacenter Scale," 2022.

[31] C. Avin, C. Griner, I. Salem, and S. Schmid, "An Online Matching Model for Self-Adjusting ToR-to-ToR Networks," 2020.

[32] C. Griner, J. Zerwas, A. Blenk, S. Schmid, M. Ghobadi, and C. Avin, "Cerberus: The power of choices in datacenter topology design (a throughput perspective)," in *Proc. ACM SIGMETRICS*, 2022.

[33] E. Feder, I. Rathod, P. Shyamsukha, R. Sama, V. Aksenov, I. Salem, and S. Schmid, "Brief Announcement: Toward Self-Adjusting Networks for the Matching Model," in *33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2021.

[34] C. Avin and S. Schmid, "Renets: Statically-optimal demand-aware networks," in *Proc. SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, 2021.

[35] "Polatis Series 6000," www.polatis.com, [Online; accessed 10-May-2022].

[36] B. Peres, O. Souza, O. Goussevskaia, S. Schmid, and C. Avin, "Distributed self-adjusting tree networks," in *Proc. IEEE INFOCOM*, 2019.

[37] D. Sleator and R. Tarjan, "Self-adjusting binary search trees," *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 652–686, 1985.

[38] Y. Afek, H. Kaplan, B. Korenfeld, A. Morrison, and R. E. Tarjan, "CBTree: A Practical Concurrent Self-adjusting Search Tree," in *Proc. DISC*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 1–15.

[39] O. A. d. O. Souza, O. Goussevskaia, and S. Schmid, "CBNet: Minimizing Adjustments in Concurrent Demand-Aware Tree Networks," in *IEEE Int. Parallel and Distributed Processing Sym. (IPDPS)*, 2021, pp. 382–391.

[40] E. Feder, I. Rathod, P. Shyamsukha, R. Sama, V. Aksenov, I. Salem, and S. Schmid, "Lazy Self-Adjusting Bounded-Degree Networks for the Matching Model," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2022.

[41] caiocaldeira3, "Opticnet framework (source code)," https://github.com/caiocaldeira3/OpticalNet, 2024, accessed 21-February-2024.

[42] ——, "Semi Displaynet in OpticNet (source code)," https://github.com/caiocaldeira3/SemiDisplayOpticNet, 2024, accessed 21-February-2024.

[43] ——, "CBNet in OpticNet (source code)," https://github.com/caiocaldeira3/CBOpticalNet, 2024, accessed 21-February-2024.

[44] ——, "Displaynet in OpticNet (source code)," https://github.com/caiocaldeira3/DisplayOptNet, 2024, accessed 21-February-2024.

[45] "ProjecToR dataset," www.microsoft.com/en-us/research/project/projector-agile-reconfigurable-data-center-interconnect, 2016.

[46] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proceedings of the 2015 ACM SIGCOMM*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 123–137.

[47] D. Amir, T. Wilson, V. Shrivastav, H. Weatherspoon, R. Kleinberg, and R. Agarwal, "Optimal oblivious reconfigurable networks," in *Proc. of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, 2022.

[48] J. Kulkarni, S. Schmid, and P. Schmidt, "Scheduling opportunistic links in two-tiered reconfigurable datacenters," in *33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2021.

[49] C. Avin, A. Hercules, A. Loukas, and S. Schmid, "rDAN: Toward robust demand-aware network designs," in *Information Processing Letters (IPL)*, 2018.

[50] S. B. Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate carathéodory theorems," *Queueing Systems*, vol. 88, no. 3-4, pp. 311–347, 2018.

[51] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 447–458, Aug. 2013.

[52] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: a topology malleable data center network," in *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2010.

[53] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-Through: Part-time optics in data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 327–338, 2011.

[54] C. Avin, K. Mondal, and S. Schmid, "Demand-aware network design with minimal congestion and route lengths," in *Proc. IEEE INFOCOM*, 2019.

[55] J. Zerwas, C. Avin, S. Schmid, and A. Blenk, "ExRec: Experimental Framework for Reconfigurable Networks Based on Off-the-Shelf Hardware," in *Proc. of the Symposium on Architectures for Networking and Communications Systems*, ser. ANCS, 2021, p. 66–72.

Caio Alves Caldeira Caio Alves Caldeira is a graduate student at the Department of Computer Science at Universidade Federal de Minas Gerais (UFMG), Brazil. He has many interests in computer science, such as competitive programming, distributed networks, parallel programming, data science, logic and proof methods.

Otávio Augusto de Oliveira Souza Otávio Augusto de Oliveira Souza received the M.Sc degree in Computer Science in 2020 from Universidade Federal de Minas Gerais (UFMG), Brazil, where he is currently a PhD candidate in Computer Science. His research interest is in computer networking.

Olga Goussevskaia Olga Goussevskaia is an associate professor of computer science at Universidade Federal de Minas Gerais (UFMG), Brazil. She received her PhD in computer science in 2009 from ETH Zurich, Switzerland. Her main research interests include modeling, algorithm design and analysis in communication networks.

Stefan Schmid Stefan Schmid is a full professor at the Technical University of Berlin, Germany. He received his MSc (2004) and PhD (2008) from ETH Zurich, Switzerland. His research interests revolve around the fundamental algorithmic problems of networked and distributed systems.